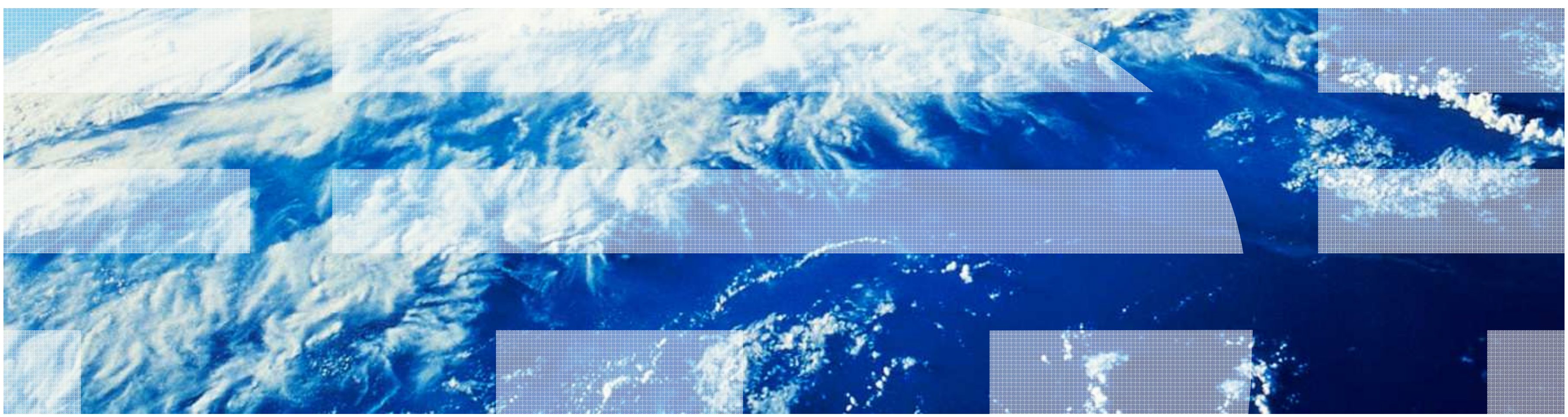
Robert W. Wisniewski

Chief Software Architect

Blue Gene Supercomputer Research

~~Out with the old,~~
~~in with the new~~

In with the old,
“bring on” the new



Obligatory Exascale Swim Lanes Slide

Outline

- **Review of exascale codesign center projects and NNSA**
  - Considerable focus remains on traditional programming paradigms
- **Challenge of introducing new models**
  - Applications want to know what new model is
  - System software and hardware want to know what models applications could use
- **Addressing new challenges while providing familiar models**
  - Communication
  - Execution
  - Reliability
- **Two codesign messages**
  - Model: IP and SW API
  - Application questions
- **Conclusion**

Codesign slide 1: Energy and Transportation

WG 3.1 First Experts Meeting Résumé

EESI

Common main issues to be addressed (where to spend the money)

- Scalable program ,
  - Strong and weak scalability, load balancing, flexibility
  - Numerical analysis, algebra, time scales,
- Legacy code, open source,
  - Standards (MPI, OpenMP, C++, Fortran, ...)
- Mesh generation
- Coupling multi physics codes with efficiency

Data management (sorting memory for fast access, allocating new memory as needed in smaller chunks, treat parts of memory that are rarely/never needed based on heuristic algorithms, ...)

- Particle simulation
- Human resources, training (what level?)
- Toward "defensive programming" ...

Codesign slide 2: Fundamental Science

WG3.3

EESI

- Preparing for the next steps: **Short term perspectives**
  - Modularisation of codes
    - share components of codes between different groups
  - Most groups start thinking in terms of
    - extending codes to hybrid:
      MPI + OpenMP / P-Threads
    - writing codes or parts of codes for:
      GPU
    - planning extensions for codes in multi-stage parallelism
      MPI + OpenMP + accelerator (GPU, FPGA,...)

Partner Logo

Codesign slide 3: Nuclear Engineering

CESAR - Reactor Core Physics

Nek: Incompressible CFD/ Conjugate Heat Transfer

UNIC: Neutral Particle Transport

Diablo: Thermo-mechanics

TRIDENT

- All three building blocks: Nek, UNIC, Diablo, have had some success at petascale
- Success leans heavily on certain properties
  - -All three Fortran/C+MPI, no threading (good enough to run cores as MPI procs)
  - -For performance all three depend on
    - -sufficient memory per MPI process
    - -highly efficient MPI collectives
    - -for single proc performance, sufficient memory bandwidth
    - -optimized matrix vector products
    - -good tools to analyze performance (both use of mem hierarchy and MPI ops)

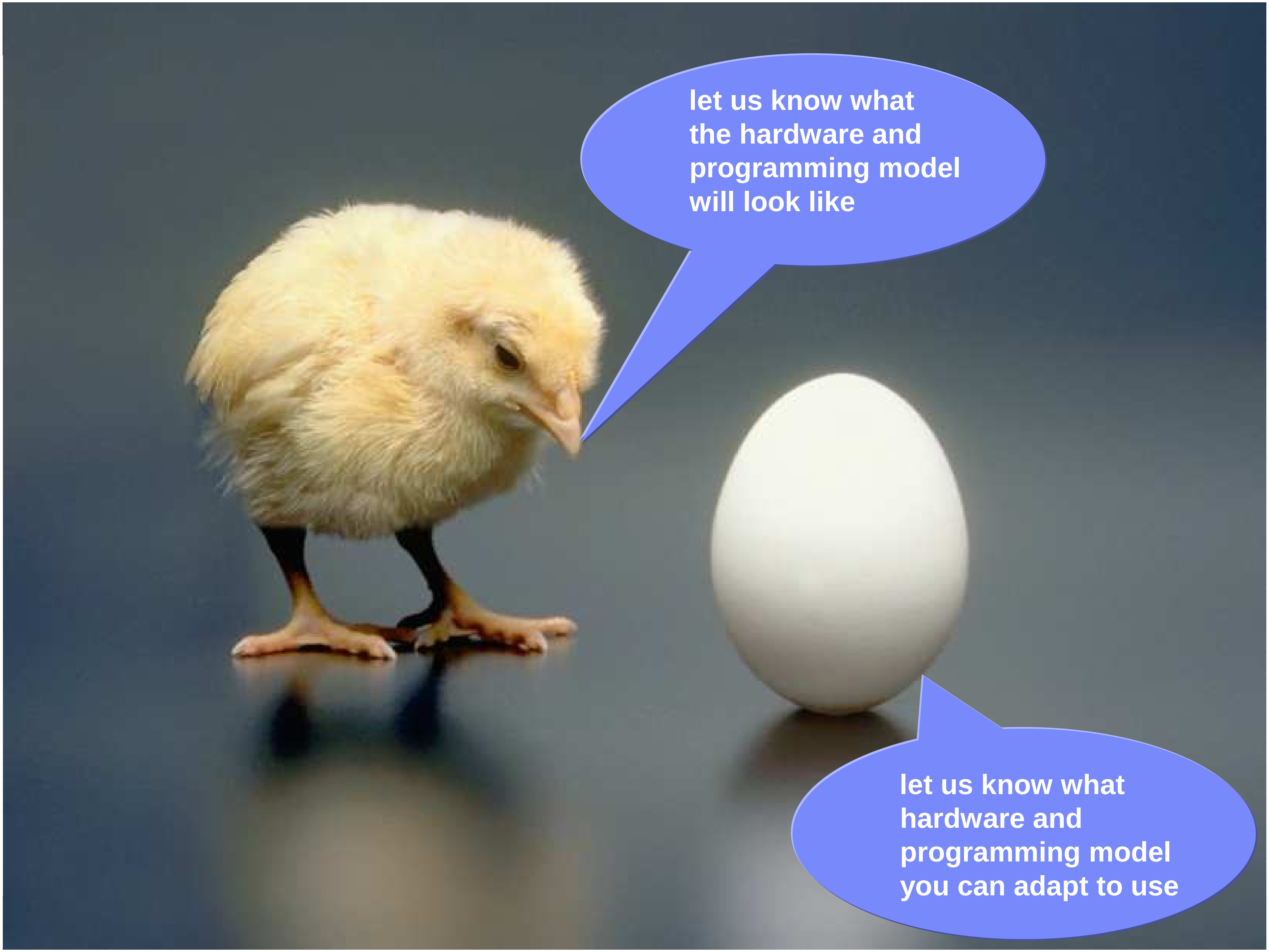
Codesign slide 4: Fusion

What the Fusion Codes need for Exascale

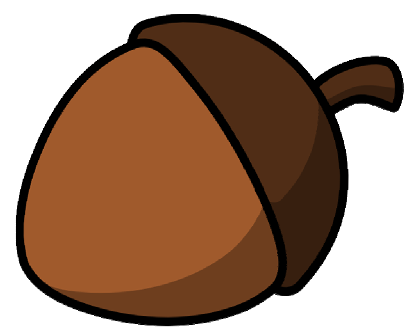
- Whole application coupling using FACETS (or other) enabled by OS
- Math Research
  - Variety of TOPS scalable libraries (e.g., PETSc, SUNDIALS, SuperLU)
  - Linear solvers, nonlinear solvers, time integration
  - Communication-avoiding and latency-tolerant algorithms
- Programming Model Research – see examples of OpenMP tasking and mixed CAF/MPI/OpenMP code
- Enabling tools (e.g. ROSE for skeleton extraction and possible automatic hybridization)
- Architecture Research
  - HW simulators for exascale designs
- Tools that work with our mixed programming model codes
- UQ Analysis especially in connection with experimental data
- Data management and I/O support for full simulations and visualization

NNSA Meeting on From Petascale to Exascale

- NNSA's meeting initiating public interaction on exascale
  - Programming models: MPI and OpenMP


let us know what the hardware and programming model will look like
let us know what hardware and programming model you can adapt to use

?

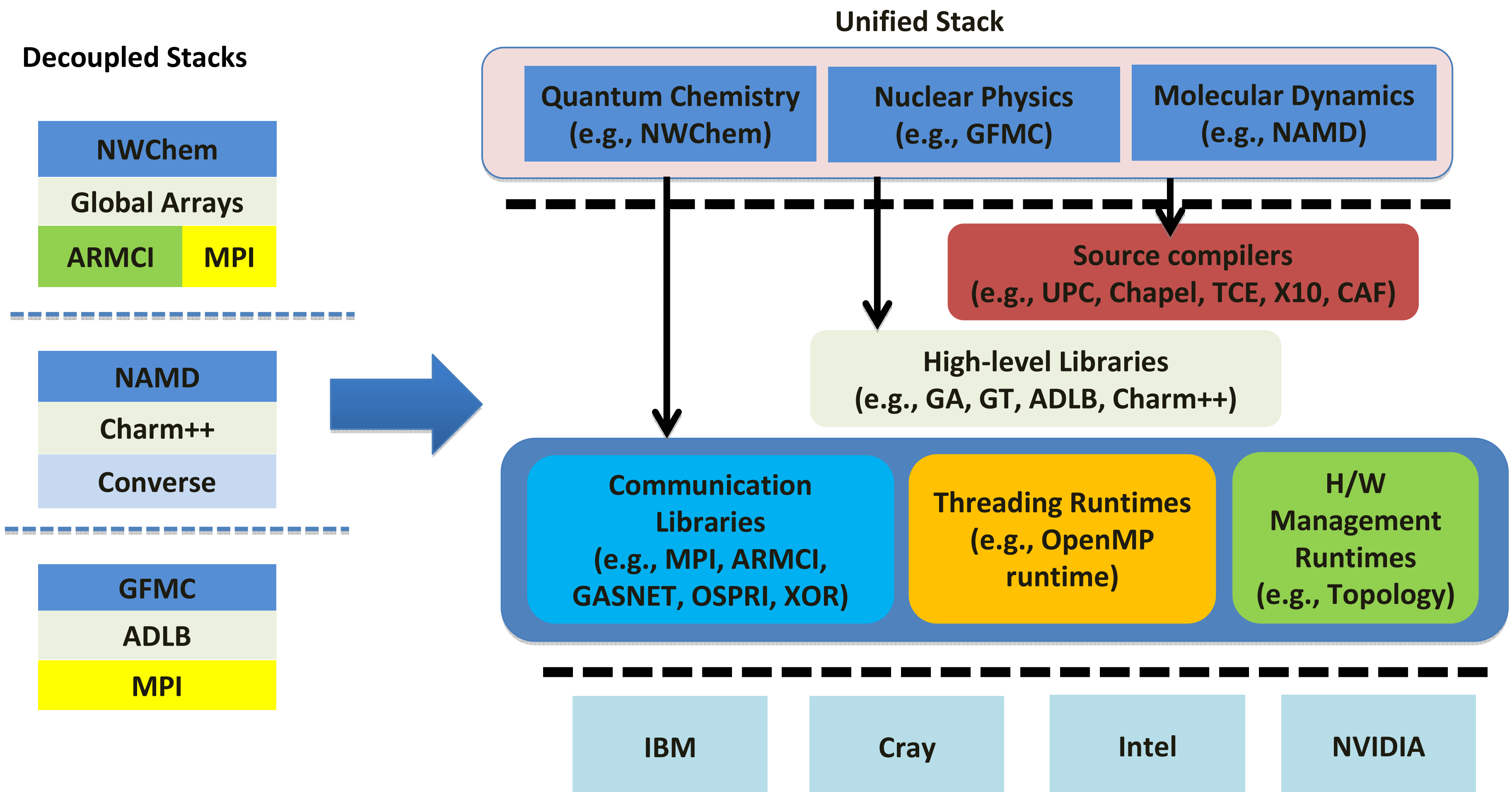


Goal

- **Run Legacy Applications and Existing Models**
  while
  Providing Applications with New Models and Latest Capabilities

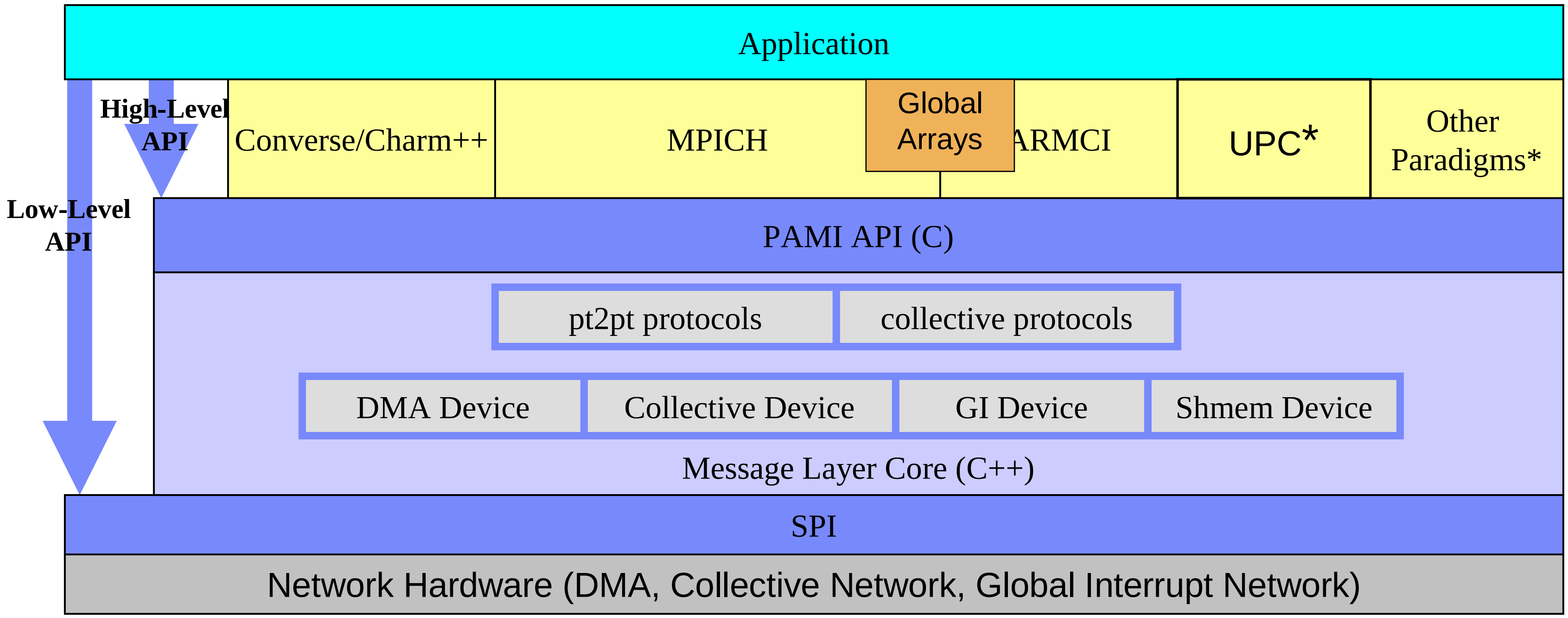
- **Three Primary Areas**
  – **Communication and runtime**
  – **Execution and system mechanisms**
  – **Reliability**

Programming Models and Runtime Systems



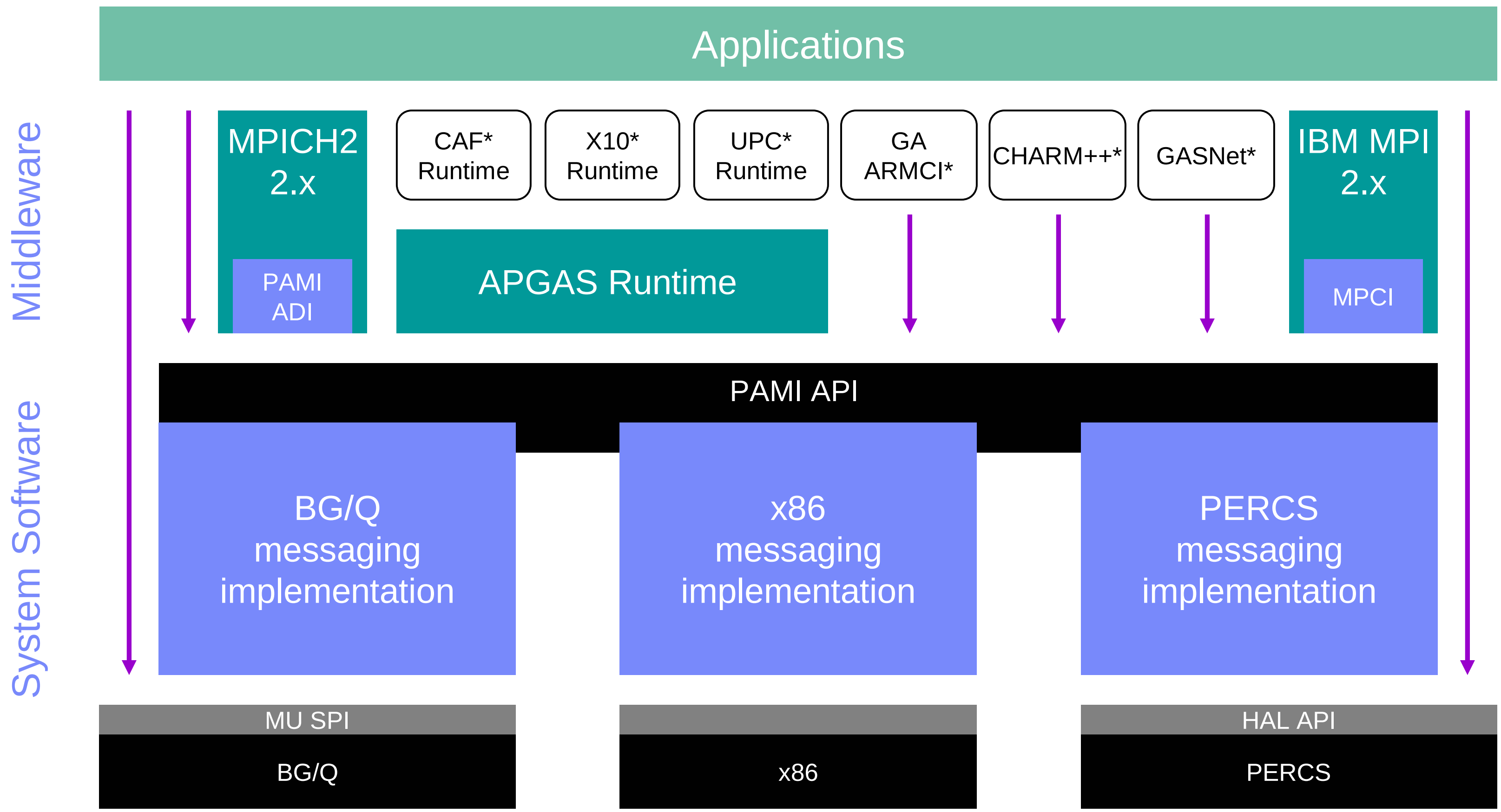
The key is to provide a unified architecture with multiple levels of capabilities and ALLOW APPLICATIONS TO BREAK THE LAYERING→ transition path for applications!

Supporting Different Shared Memory/Threading Models on PAMI Can Have Familiar Programming Models for Exascale



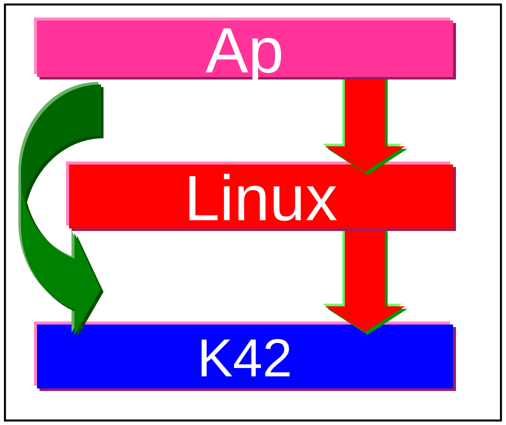
- **Message Layer Core has C++ message classes and other utilities to program different network devices**
- **Support many programming paradigms**

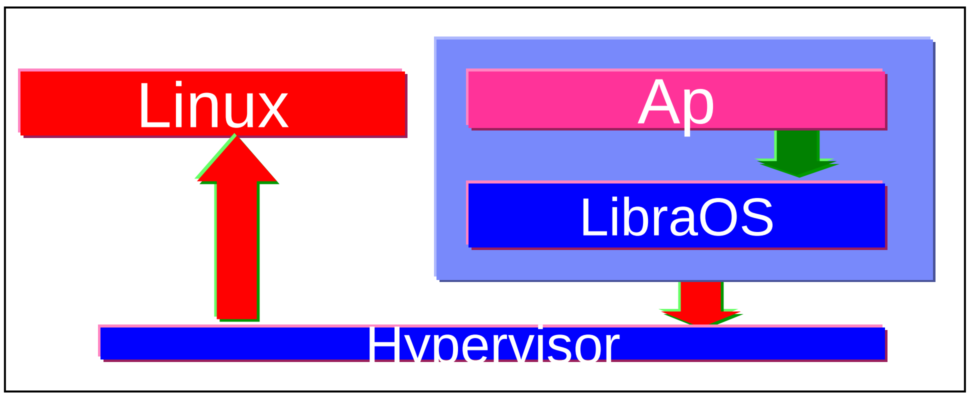
Supporting Different Shared Memory/Threading Models on PAMI Can Have Familiar Programming Models for Exascale

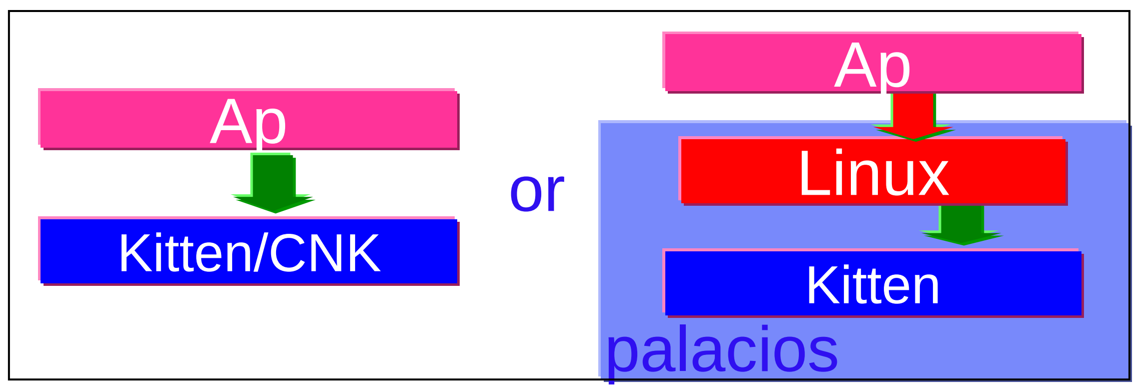


*all runtimes are not supported on all platforms

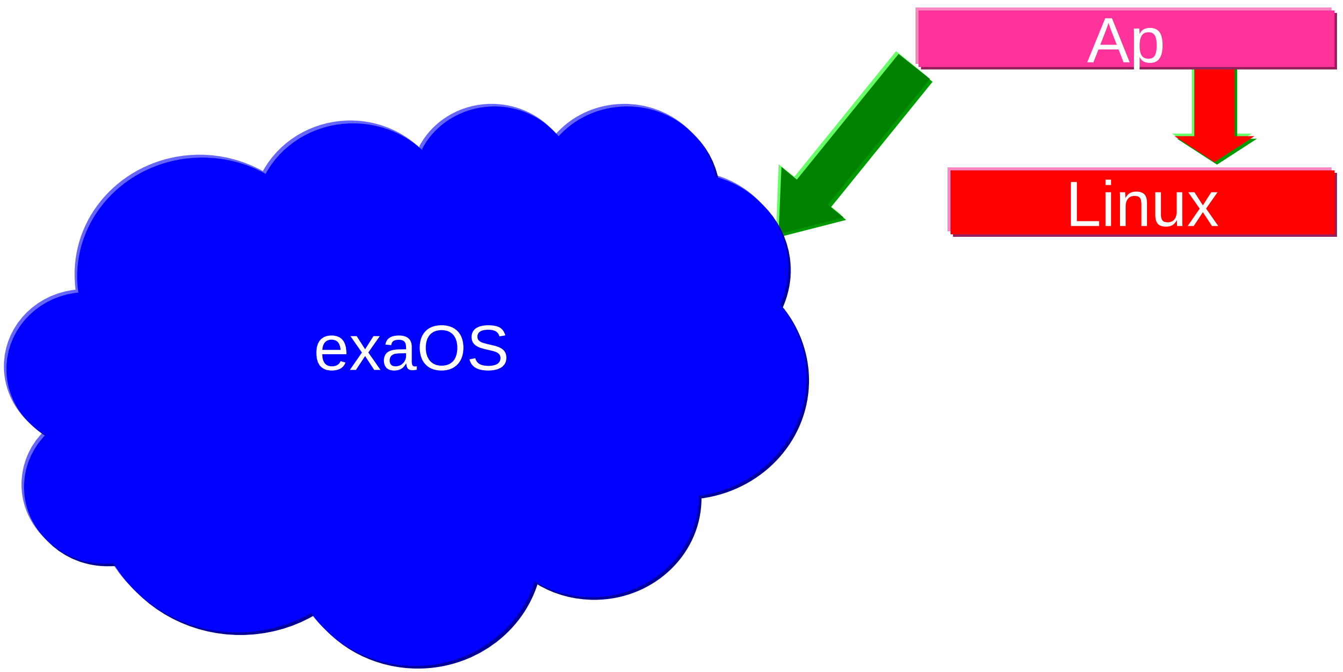
Progression of Execution Approaches







Logical End of Progression is Each Service As Needed

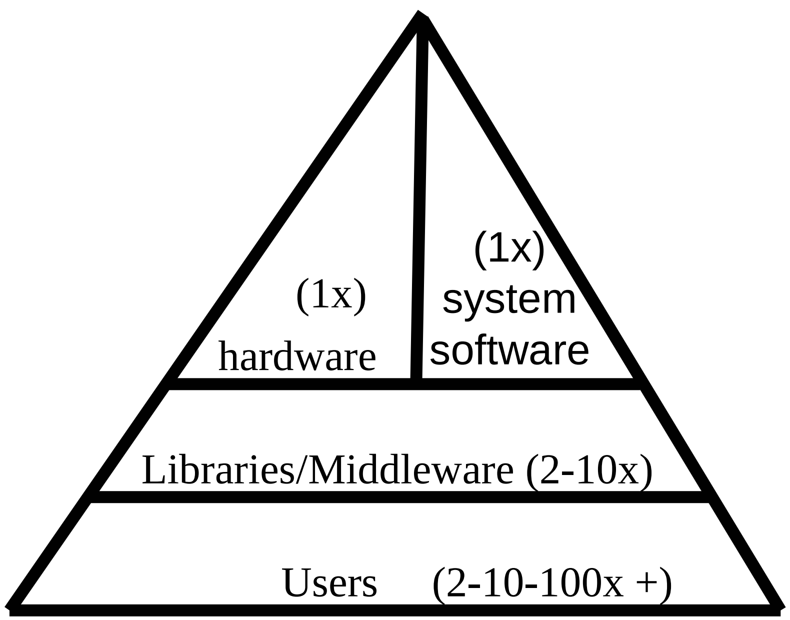


Reliability and impact on users – higher up pyramid less user impact

- Amount of silicon will make reliability a more difficult challenge
- Multiple potential directions: work with the user community to determine

  Direction 1) Push hardware and system software to guarantee correctness

  Direction 2) Leave it to the users to deal with hardware faults

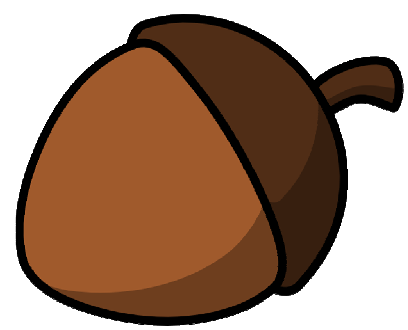


•Key to scalability is to keep it simple and predictable

•Keep reliability complexity away from the user as that is where the real cost is

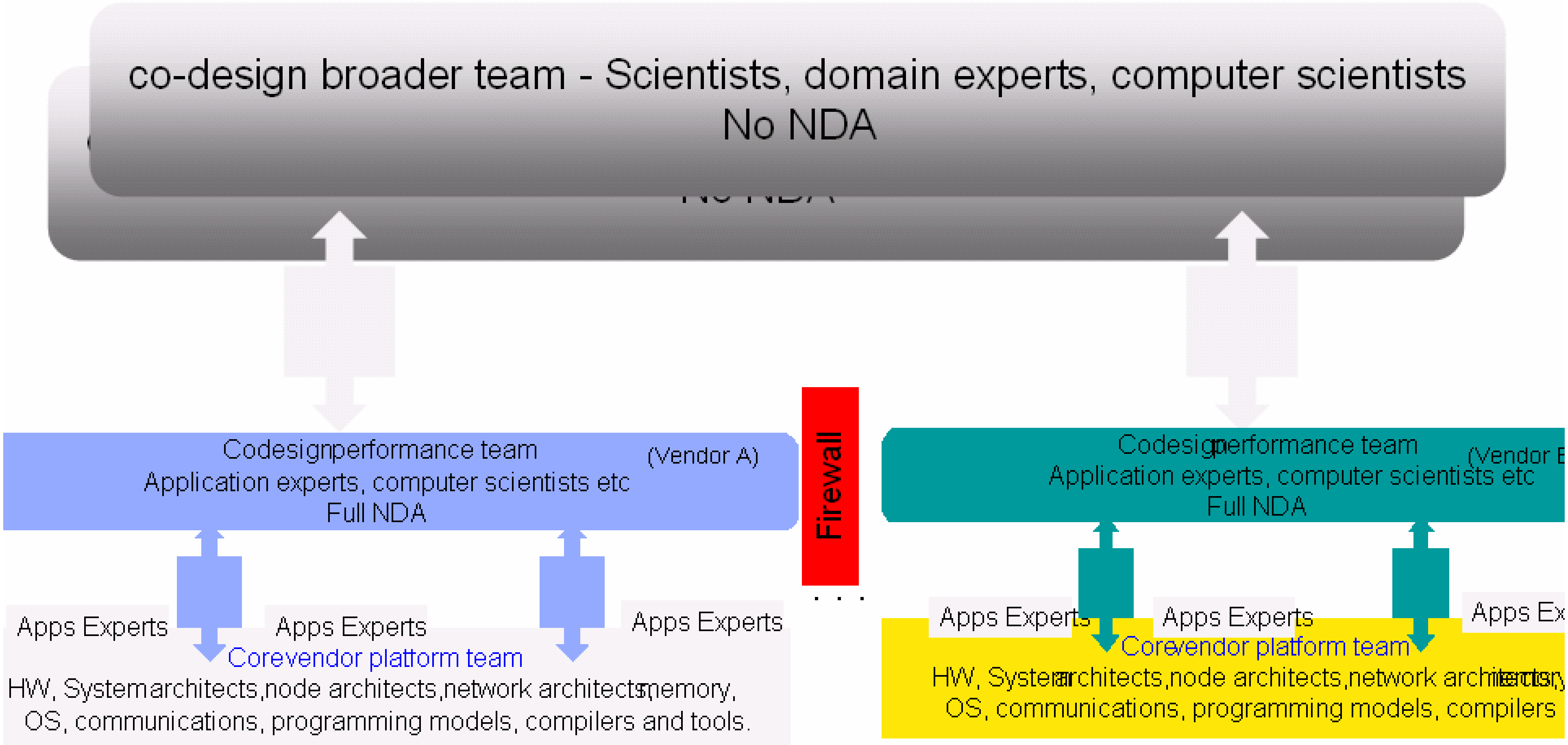
•Use hardware/software to perform local recovery at system level

?

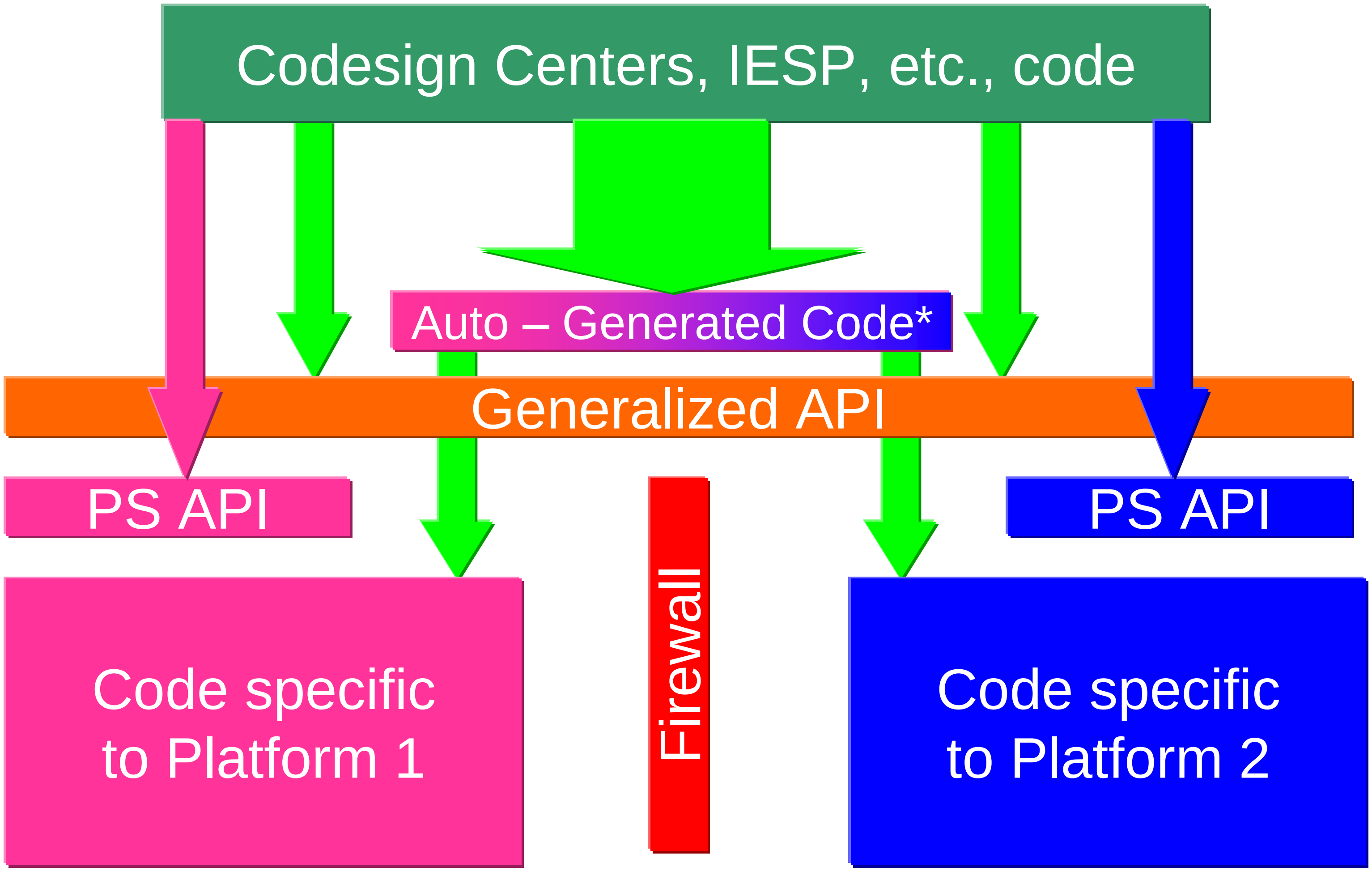


- Communication model allowing MPI to be combined with newer models such as UPC, GA, Charm++, X10, Chapel, etc
- Execution model allowing known efficient LWK mechanisms while allowing generic Linux calls
- Reliability model that assumes a significant role of the hardware and system software not placing full burden on the application

'Virtual' co-design center - proposed structure



One Model How This Might Work



*idea from Chemistry Exascale Codesign Center – thanks to Robert Harrison

Application Information Useful to Architectural Decisions (Highlights)

- Bottleneck identification
  - Important to identify the most crucial bottleneck to application, can not provide more of everything
- Computation
  - How much of your application is amenable to running on cores that are largely computation
  - How much of your application can not be parallelized, i.e., inherently serial
  - How valuable is branch prediction, Out Of Order, etc.
  - How many threads per MPI tasks: would you want, could you utilize
- Memory:
  - How well can you predict memory access pattern
  - How much memory does your node need: minimum requirements, threshold points
  - How is the memory used, read only, how widely accessed, how much redundant state
  - For a 1TF node what point does memory bandwidth become the bottleneck
  - How sensitive is your application to NUMA
- Resilience
  - How important is resilience to your application
  - How much and what character of faults can be handled in the application
  - How much are you willing to pay in performance for underlying hardware and system to handle resilience
- Communication and Synchronization
  - What types are used with what frequency, e.g., allreduce, barrier, pt-to-pt
- I/O and storage
  - What are the inherent challenges versus those caused by current implementation
- Control and resource management
  - How would you utilize an exascale machine, job mix, capability versus capacity, etc.

We will be able to bridge the gap between disparate architectures by providing support for running legacy applications while allowing new models to fully leverage the hardware. This can be accomplished with innovative evolution to existing communication, execution, and reliability models.

(we should invest in, but not rely on, revolutionary approaches in targeted areas)